SPYWOLF

Security Audit Report

Audit prepared for

Nudes.AI

Completed on

April 06, 2024

KEY RESULTS

Cannot mint new tokens	Passed
Cannot pause trading (honeypot)	Passed
Cannot blacklist an address	Passed
Cannot raise taxes over 25%?	Passed
No proxy contract detected	Passed
Not required to enable trading	Passed
No hidden ownership	Passed
Cannot change the router	Passed
No cooldown feature found	Passed
Bot protection delay is lower than 5 blocks	Passed
Cannot set max tx amount below 0.05% of total supply	Passed
The contract cannot be self-destructed by owner	Passed

For a more detailed and thorough examination of the heightened risks, refer to the subsequent parts of the report.

N/A = Not applicable for this type of contract

*Only new deposits/reinvestments can be paused

OVERVIEW

This goal of this report is to review the main aspects of the project to help investors make an informative decision during their research process.

You will find a a summarized review of the following key points:

- ✔ Contract's source code
- ✔ Owners' wallets
- ✔ Tokenomics
- ✔ Team transparency and goals
- ✔ Website's age, code, security and UX
- ✔ Whitepaper and roadmap
- ✔ Social media & online presence

The results of this audit are purely based on the team’s evaluation and does not guarantee nor reflect the projects outcome and goal

- SPYWOLF Team -

TABLE OF CONTENTS

NUDES.AI

PROJECT DESCRIPTION

According to their website:

Nudes.ai is a blockchain-powered adult entertainment platform designed to offer a unique user experience. Users can chat and interact with dream companions. Users can also opt to BECOME the characters complete with personal pages.

Release Date: TBD

Category: Adult/AI

$NLG
Governance Token

Token Name
NeverLetGo

Symbol
NLG

Contract Address
N/A

Network
N/A

Language
Solidity

Deployment Date
N/A

Contract Type
Token without taxes

Total Supply
100,000

Status
Not launched

TAXES

Buy Tax
none

Sell Tax
none

*Taxes cannot be changed

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

VULNERABILITY ANALYSIS

ID	Title	
SWC-100	Function Default Visibility	**Passed**
SWC-101	Integer Overflow and Underflow	**Passed**
SWC-102	Outdated Compiler Version	**Passed**
SWC-103	Floating Pragma	**Passed**
SWC-104	Unchecked Call Return Value	**Passed**
SWC-105	Unprotected Ether Withdrawal	**Passed**
SWC-106	Unprotected SELFDESTRUCT Instruction	**Passed**
SWC-107	Reentrancy	**Passed**
SWC-108	State Variable Default Visibility	**Passed**
SWC-109	Uninitialized Storage Pointer	**Passed**
SWC-110	Assert Violation	**Passed**
SWC-111	Use of Deprecated Solidity Functions	**Passed**
SWC-112	Delegatecall to Untrusted Callee	**Passed**
SWC-113	DoS with Failed Call	**Passed**
SWC-114	Transaction Order Dependence	**Passed**
SWC-115	Authorization through tx.origin	**Passed**
SWC-116	Block values as a proxy for time	**Passed**
SWC-117	Signature Malleability	**Passed**
SWC-118	Incorrect Constructor Name	**Passed**

VULNERABILITY ANALYSIS

ID	Title	
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed

VULNERABILITY ANALYSIS

NO ERRORS FOUND

MANUAL
CODE REVIEW

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time.

We categorize these vulnerabilities by 4 different threat levels.

THREAT LEVELS

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance, functionality and should be fixed before moving to a live environment.

Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Contract's deployer can withdraw token balances via the withdrawRemainingAllocation() for up to 57,500 tokens. Withdrawable amount increases with 175 tokens each 24 hours.

Note:
Contract's token balances can be withdrawn at any time as the function does not check how many withdrawals should occur per day.

```
function withdrawRemainingAllocation() external onlyDeployer {
    require(_remainingAllocation > 0, "NLGToken: no remaining allocation left");

    // Calculate the rate of tokens released per second
    uint256 tokensPerSecond = uint256(175 * 10**18) / (24 * 60 * 60); // 175 tokens released every 24 hours

    // Calculate the amount of tokens to release based on elapsed time
    uint256 elapsedSeconds = block.timestamp - _vestingStartTime;
    uint256 tokensToRelease = elapsedSeconds * tokensPerSecond;

    // Ensure the amount to release does not exceed the remaining allocation
    tokensToRelease = min(tokensToRelease, _remainingAllocation);

    // Update remaining allocation and transfer tokens to the deployer
    _remainingAllocation -= tokensToRelease;
    _transfer(address(this), _deployer, tokensToRelease);

    emit WithdrawRemainingTokens(tokensToRelease);
}
```

FOUND THREATS

Informational

Contract's deployer can withdraw token balances via withdrawOwnerAllocation() for up to 10,000 tokens.
Up to 1667 tokens can be withdrawn for 3 month period, at least 3 months after contract's initial deployment date.

```
function withdrawOwnerAllocation() external onlyDeployer {
    require(_ownerAllocation > 0, "NLGToken: no owner allocation left");

    uint256 elapsedSeconds = block.timestamp - _vestingStartTime;
    uint256 threeMonthsInSeconds = 90 days; // 90 days in seconds

    // Check if the 90-day cliff has passed
    require(elapsedSeconds >= threeMonthsInSeconds, "NLGToken: 90-day cliff has not passed yet");

    // Convert seconds to days after the cliff
    uint256 elapsedDaysAfterCliff = (elapsedSeconds - threeMonthsInSeconds) / 1 days;
    // Calculate the number of 90-day periods that have passed
    uint256 numberOfPeriods = elapsedDaysAfterCliff / 90;

    // Calculate the total tokens available in this window
    uint256 tokensAvailable = numberOfPeriods * 1667 * 10**18; // 6 quarters of vesting

    // Calculate the tokens to release in this window
    uint256 tokensToRelease = tokensAvailable - _tokensClaimed;

    // Ensure the amount to release is greater than zero
    require(tokensToRelease > 0, "NLGToken: no tokens available for release");

    // Ensure the amount to release does not exceed the owner allocation
    tokensToRelease = min(tokensToRelease, _ownerAllocation);

    // Update tokens claimed by the owner
    _tokensClaimed += tokensToRelease;

    // Update owner allocation
    _ownerAllocation -= tokensToRelease;

    // Transfer tokens to the owner
    _transfer(address(this), _deployer, tokensToRelease);

    emit WithdrawOwnerTokens(tokensToRelease);
}
```

$NUDES
Utility Token

Token Name NudesAI	Symbol NUDES
Contract Address N/A	
Network N/A	Language Solidity
Deployment Date N/A	Contract Type Token with taxes
Total Supply 100,000,000	Status Not launched

TAXES

Buy Tax	Sell Tax
5%	5%

*Taxes can be changed in future

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Owner can set buy and sell fees up to 5%.
Combined buy+sell = 10%.
When fees are above 0, there will be certain amount of tokens that will be deducted from every transaction that users make.
Deducted amount will be as much as the fees % from total amount that user had bought, sold and/or transferred.

```
function setTransferTax(uint256 newTaxRate) external onlyDeployer {
    require(newTaxRate <= 5, "NUDESToken: Tax rate must be between 0 and 5 inclusive");
    _transferTax = newTaxRate;
    emit TransferTaxUpdated(newTaxRate);
}
```

Owner can exclude address from fees.
When address is excluded from fees, the user will receive the whole amount of the bought, sold and/or transferred tokens.

```
function addTaxExemption(address exemptAddress) external onlyDeployer {
    _isTaxExempt[exemptAddress] = true;
    emit TaxExemptionAdded(exemptAddress);
}
```

FOUND THREATS

Informational

Contract's deployer can withdraw token balances via the withdrawTreasuryAllocation() for up to 20,000,000 tokens. Withdrawable amount increases with 54,000 tokens each 24 hours. Contract's token balances can be withdrawn at any time as the function does not check how many withdrawals should occur per day.

```
function withdrawTreasuryAllocation() external onlyDeployer {
    require(_treasuryAllocation > 0, "NUDESToken: no treasury allocation left");

    // Calculate the rate of tokens released per secon0d
    // 54,000 tokens released every 24 hours
    uint256 tokensPerSecond = uint256(54_000 * 10**18) / (24 * 60 * 60);

    // Calculate the amount of tokens to release based on elapsed time
    uint256 elapsedSeconds = block.timestamp - _vestingStartTime;
    uint256 tokensToRelease = elapsedSeconds * tokensPerSecond;

    // Ensure the amount to release does not exceed the treasury allocation
    tokensToRelease = min(tokensToRelease, _treasuryAllocation);

    // Update treasury allocation and transfer tokens to the deployer
    _treasuryAllocation -= tokensToRelease;
    _transfer(address(this), _deployer, tokensToRelease);

    emit WithdrawTreasuryTokens(tokensToRelease);
}
```

FOUND THREATS

Informational

Contract's deployer can withdraw token balances via withdrawTeamAllocation() for up to 10,000,000 tokens.
Up to 1,666,666 tokens can be withdrawn for 3 month period, at least 3 months after contract's initial deployment date.

```
function withdrawTeamAllocation() external onlyDeployer {
    require(_teamAllocation > 0, "NUDESToken: no team allocation left");

    uint256 elapsedSeconds = block.timestamp - _vestingStartTime;
    uint256 threeMonthsInSeconds = 90 days; // 90 days in seconds

    // Check if the 90-day cliff has passed
    require(elapsedSeconds >= threeMonthsInSeconds, "NUDESToken: 90-day cliff has not passed yet");
    // Convert seconds to days after the cliff
    uint256 elapsedDaysAfterCliff = (elapsedSeconds - threeMonthsInSeconds) / 1 days;

    // Calculate the number of 90-day periods that have passed
    uint256 numberOfPeriods = elapsedDaysAfterCliff / 90;

    // Calculate the total tokens available in this window
    uint256 tokensAvailable = numberOfPeriods * 1666666 * 10**18;

    // Calculate the tokens to release in this window
    uint256 tokensToRelease = tokensAvailable - _tokensClaimed;

    // Ensure the amount to release is greater than zero
    require(tokensToRelease > 0, "NUDESToken: no tokens available for release");

    // Ensure the amount to release does not exceed the team allocation
    tokensToRelease = min(tokensToRelease, _teamAllocation);

    // Update tokens claimed by the team
    _tokensClaimed += tokensToRelease;

    // Update team allocation
    _teamAllocation -= tokensToRelease;

    // Transfer tokens to the team
    _transfer(address(this), _deployer, tokensToRelease);

    emit WithdrawTeamTokens(tokensToRelease);
}
```

$NUDES Airdropper

Token Name
N/A

Symbol
N/A

Contract Address
N/A

Network
N/A

Language
Solidity

Deployment Date
N/A

Contract Type
Airdropper

Total Supply
N/A

Status
Not launched

TAXES

Buy Tax
none

Sell Tax
none

*Taxes can be changed in future

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Owner can create airdrops for any amounts to any addresses.

```
function createAirdrop(uint256[] memory _amounts, address[] memory _recipients) external onlyOwner {
    require(_amounts.length == _recipients.length, "Mismatch between amounts and recipients");

    uint256 airdropId = nextAirdropId++;
    for (uint256 i = 0; i < _recipients.length; i++) {
        airdrops[airdropId][_recipients[i]] = _amounts[i];
    }

    emit AirdropCreated(airdropId);
}
```

Users that have valid airdrops can claim them via the claim() function for up to treasury wallet's balances.

Note: If user's mapped airdrop exceeds the balances of the treasury wallet, they won't be able to claim their airdrop.

```
function claim(uint256 _airdropId) external {
    require(airdrops[_airdropId][msg.sender] > 0, "No airdrop to claim");
    require(!claimed[_airdropId][msg.sender], "Already claimed");

    uint256 amount = airdrops[_airdropId][msg.sender];
    uint256 currentAllowance = NUDES.allowance(WALLET_TREASURY, address(this));
    uint256 treasuryBalance = NUDES.balanceOf(WALLET_TREASURY);

    require(currentAllowance >= amount, "Insufficient allowance");
    require(treasuryBalance >= amount, "Insufficient balance in treasury");

    claimed[_airdropId][msg.sender] = true;

    // Use transferFrom to move tokens from the treasury wallet to the claimant
    NUDES.transferFrom(WALLET_TREASURY, msg.sender, amount);

    emit Claimed(_airdropId, msg.sender, amount);
}
```

$NGL Staking

Token Name
N/A

Symbol
N/A

Contract Address
N/A

Network
N/A

Language
Solidity

Deployment Date
N/A

Contract Type
Staking

Total Supply
N/A

Status
Not launched

TAXES

Buy Tax
none

Sell Tax
none

*Taxes can be changed in future

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Users can unstake their staked amount at any time.

Note: If large number of users exists into the stakerAddresses mapping, the necessary gas for iterations through the array can exceed the max gas limit allowed for block (block size). If that happens, transaction will revert and user will be unable to unstake.
Max gas limit for block can vary between different EVMs (eg. current gas limits per block at time of this audit -> BSC - 140M, ETH - 30M).
Consider using external instead of public modifier for gas savings and/or implement different gas efficient mechanism for users tracking.

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices/19391#19391

```
function unstake(uint256 _amount) public {
    require(stakes[msg.sender] >= _amount, "Insufficient stake");

    // Reduce the staked amount for the user
    stakes[msg.sender] -= _amount;
    totalStakes -= _amount;

    // Transfer the unstaked tokens back to the user
    nlgToken.transfer(msg.sender, _amount);

    // Emit the Unstaked event
    emit Unstaked(msg.sender, _amount);

    // If the user has unstaked their full amount, remove them from stakerAddresses
    if (stakes[msg.sender] == 0) {
        for (uint256 i = 0; i < stakerAddresses.length; i++) {
            if (stakerAddresses[i] == msg.sender) {
                // Replace the address to be removed with the last address in the array
                stakerAddresses[i] = stakerAddresses[stakerAddresses.length - 1];
                // Remove the last element from the array
                stakerAddresses.pop();
                break;
            }
        }
    }
}
```

Team NFTs

Token Name TeamNFT	Symbol TNFT
Contract Address N/A	
Network N/A	Language Solidity
Deployment Date N/A	Contract Type NFT
Total Supply N/A	Status Not launched

TAXES

Buy Tax	Sell Tax
none	**none**

*Taxes can be changed in future

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

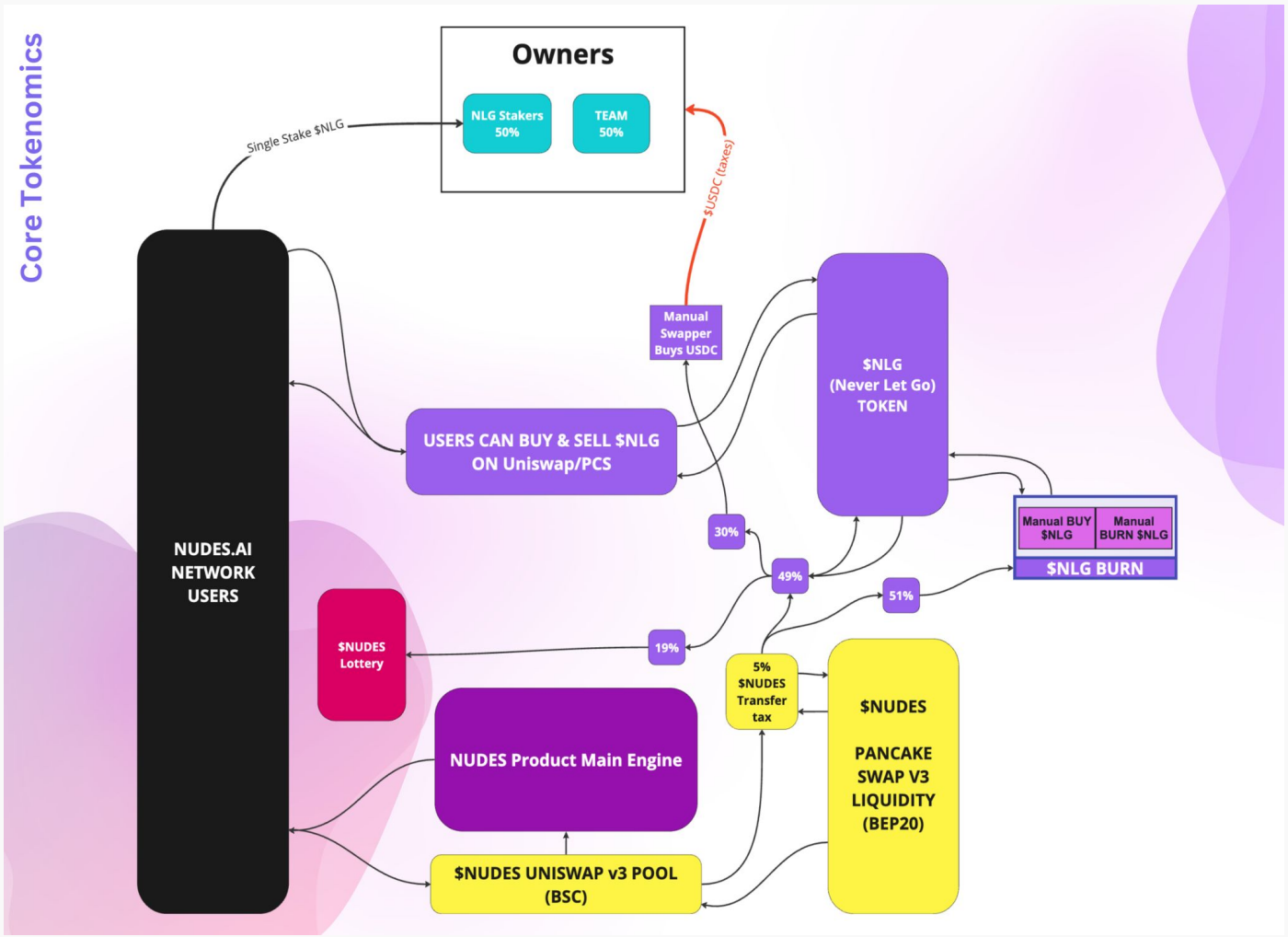
Informational

NFTs are minted on contract deploying by the deployer.

```
constructor(
    address[] memory _wallets,
    uint256[] memory _quantities,
    string[] memory _uris,
    uint256[] memory _tokenIds
) ERC1155("") {
    for (uint256 i = 0; i < _wallets.length; i++) {
        tokenURIs[_tokenIds[i]] = _uris[i];
        _mintAndUpdateHolders(_wallets[i], _tokenIds[i], _quantities[i], "");
    }
}
```

The following tokenomics are based on the project's whitepaper and/or website:

- 50% - NGL Stakers
- 50% - Team



TOKENOMICS

WEBSITE

Website URL
https://nudes.ai/

Domain Registry
https://www.namecheap.com

Domain Expiration
May 3, 2025

Technical SEO Test
Passed

Security Test
Passed. SSL certificate present

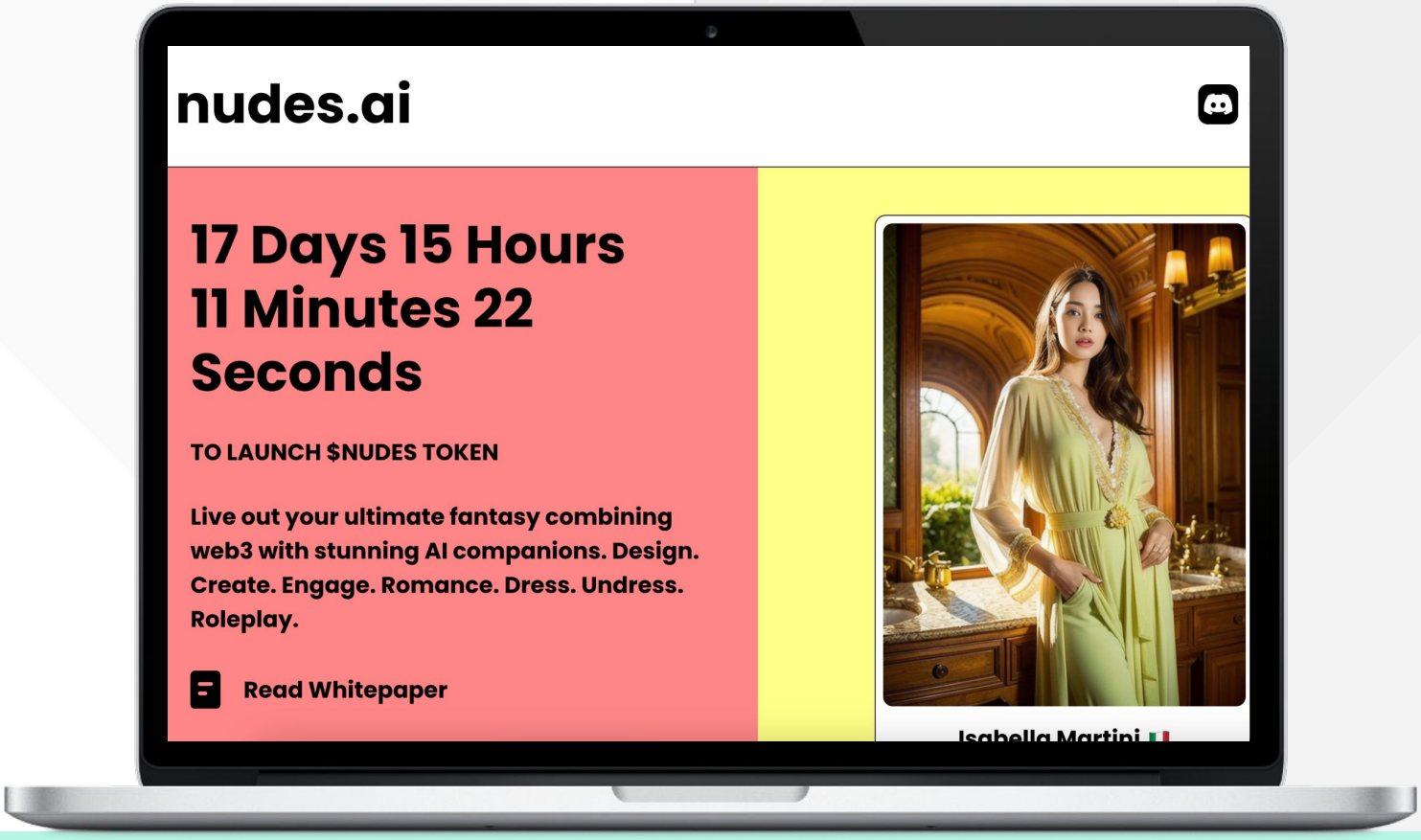
Design
Under construction at the time of audit.

Content
Under construction at the time of audit.

Whitepaper
Yes

Roadmap
Yes

Mobile-friendly?
Yes



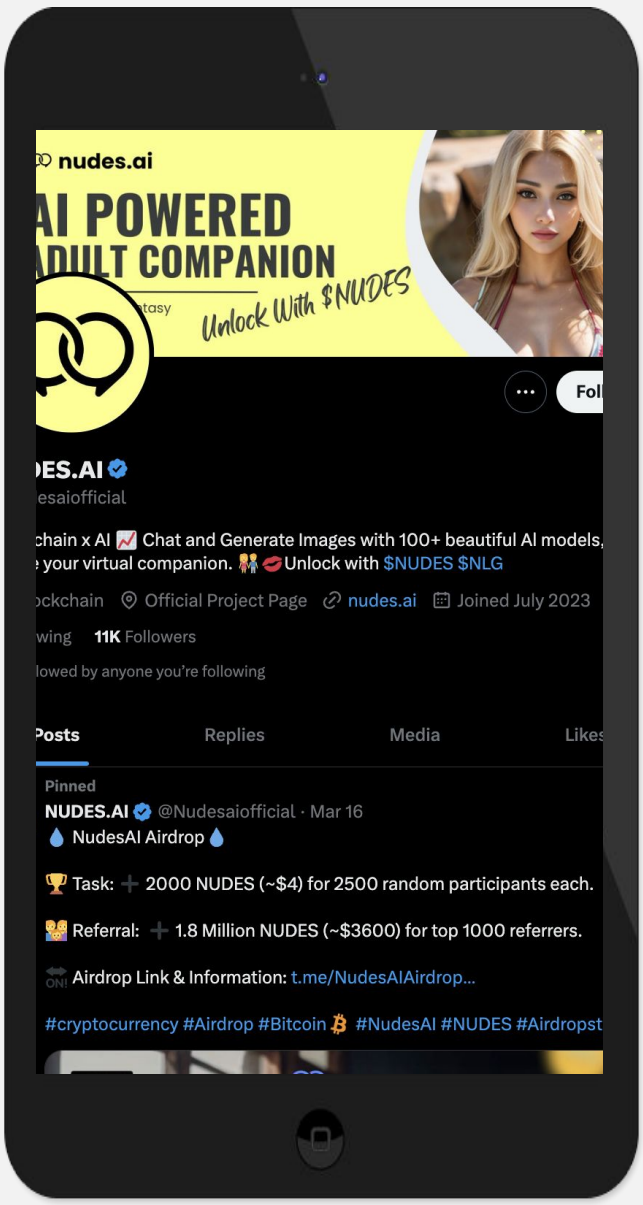
Nudes.ai (Under Construction)

SOCIAL MEDIA

& ONLINE PRESENCE

ANALYSIS

Social media presence is new but active.



Twitter's X

@Nudesaiofficial

- 11K followers
- Responds to comments
- Daily posts

Discord

/invite/TfEukaPhmN

- 4609 members
- ACtive community

Telegram

- Not available

Medium

- Not available

ABOUT US

We are a growing crypto security agency offering audits, KYCs and consulting services for some of the top names in the crypto industry.

- **OVER 700 SUCCESSFUL CLIENTS**
- **MORE THAN 1000 SCAMS EXPOSED**
- **MILLIONS SAVED IN POTENTIAL FRAUD**
- **PARTNERSHIPS WITH TOP LAUNCHPADS, INFLUENCERS AND CRYPTO PROJECTS**
- **CONSTANTLY BUILDING TOOLS TO HELP INVESTORS DO BETTER RESEARCH**

SPYWOLF

CRYPTO SECURITY

Audits | KYCs | dApps
Contract Development

To hire us, reach out to
contact@spywolf.co or
t.me/joe_SpyWolf

FIND US ONLINE

SPYWOLF.CO

@SPYWOLFNETWORK

@SPYWOLFNETWORK

Disclaimer

This report shows findings based on our limited project analysis, following good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall social media and website presence and team transparency details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report.
While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER:
By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.
No one shall have any right to rely on the report or its contents, and SpyWolf and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (SpyWolf) owe no duty of care towards you or any other person, nor does SpyWolf make any warranty or representation to any person on the accuracy or completeness of the report.
The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SpyWolf hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SpyWolf hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SpyWolf, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts, website, social media and team.
No applications were reviewed for security. No product code has been reviewed.